

## Types of Queue:-

There are four types of Queue:-

- (i) Linear Queue.
- (ii) Circular Queue
- (iii) Priority Queue
- (iv) Deque. (Double ended Queue).

### (i) Linear Queue:-

- A Linear Queue, follow the (first in first Out), much like a line in real life.
- In Linear Queue, an insertion takes place from one end while deletion occurs from another end.

Insertion  $\leftarrow$  FRONT  
Deletion  $\leftarrow$  REAR

- Linear Queue can be implemented with the help of a linear array of one dimension, that's why it is called Linear Queue.

Operations  $\rightarrow$  Enqueue  
 $\rightarrow$  Dequeue

Explanation:- (Enqueue)

Index	1	2	3	4	5
values					

Initial

FRONT = 0
Rear = 0

Insert "A"

Index	1	2	3	4	5
values	A				

Front = 1
Rear = 1

Insert "B", "C", "D"

Index	1	2	3	4	5
values	A	B	C	D	

Front = 1
Rear = 4

Insert "E"

Index	1	2	3	4	5
Values	A	B	C	D	E

Front = 1
Rear = 5

Insert "F"

Index	1	2	3	4	5
values	A	B	C	D	E

Front = 1
Rear = 5

(print "Overflow" as queue is full now)

Explanation (Dequeue):-

Index	1	2	3	4	5
values	A	B	C	D	E

Front 1
Rear 5

Delete

item = A

Index	1	2	3	4	5	Front	2
values		B	C	D	E	Rear	5

Delete

Item B

Index	1	2	3	4	5	Front	3
values			C	D	E	Rear	5

Delete

Item C

Index	1	2	3	4	5	Front	4
Values				D	E	Rear	5

Delete

Item D

Index	1	2	3	4	5	Front	5
values					E	Rear	5

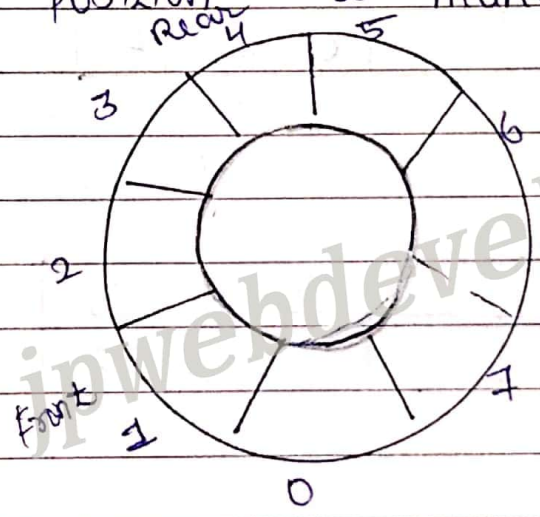
Delete

Item E

Index	1	2	3	4	5	Front	0
values						Rear	0

## 2. Circular Queue:-

- Circular Queue is a linear data structure.
- In this, operations are performed based on FIFO (first in first out) principle and last position is connected back to the first position to make a circle.



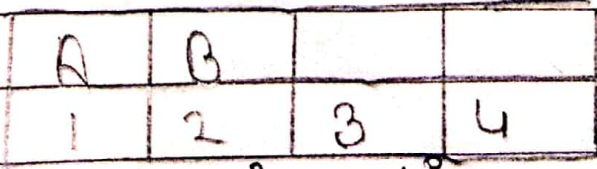
- In a normal Queue, we can insert elements until Queue becomes full. But once queue becomes full, we cannot insert the next element even if there is a space in front of queue.

FRONT = 0  
REAR = 0



(a) Initially empty Queue

FRONT = 1  
REAR = 2



(b) After insertion of A and B in queue

FRONT = 1	A	B	C	D
REAR = 4	1	2	3	4

(C) After insertion of C and D in queue

FRONT = 2		B	C	D
REAR = 4	1	2	3	4

(D) After deletion from queue

FRONT = 3			C	D
REAR = 4	1	2	3	4

(E) After deletion from queue

FRONT = 3	E		C	D
REAR = 1	1	2	3	4

(F) After insertion of E

FRONT = 4	E			D
REAR = 1	1	2	3	4

(G) After deletion from queue

FRONT = 4	E	G		D
REAR = 2	1	2	3	4

(H) After insertion of G

FRONT = 1	E	G		
REAR = 2	1	2	3	4

(I) After deletion

FRONT = 2		G		
REAR = 2	1	2	3	4

(J) After deletion from queue

FRONT = 0				
REAR = 0	1	2	3	4

(K) After deletion from queue

\_/\_/\_

\* Algorithm

CIR - QINSERT (Q, N, FRONT, REAR, ITEM),

Step 1: IF (FRONT = 1 and REAR = <sup>size of n</sup>N) OR (FRONT = REAR + 1)  
Then  
Write "overflow" and Exit.

Step 2: IF FRONT = NULL Then  
Set FRONT = 1 and REAR = 1

Else IF REAR = N Then  
Set REAR = 1

Else  
Set REAR = REAR + 1

Step 3: Set Q[REAR] = ITEM

Step 4: Exit

\_ / \_ / \_

Circular Queue deletion)

Algorithm CIR\_QDELETE(Q, N, FRONT, REAR, ITEM).

Step 1:- IF FRONT = NULL Then  
Write "Underflow" and EXIT  
[End if]

Step 2:- Set ITEM = Q [FRONT]

Step 3:- IF FRONT = REAR Then  
Set FRONT = REAR = NULL

Else if FRONT = N Then  
Set FRONT := 1

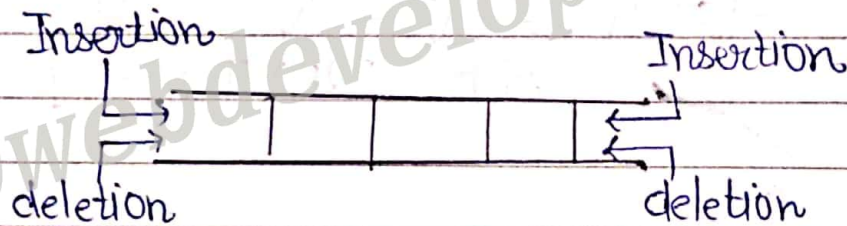
Else  
Set FRONT = FRONT + 1

[End IF]

Step 4:- EXIT

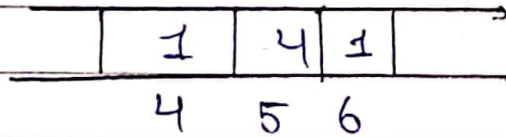
### 3. DEQUE (Double Ended Queue): -

- In this insertion and deletion can be performed at both end i.e. front pointer can be used for insertion and rear pointer can be used for deletion.
- elements can be added and removed at both ends (front and rear), but not in middle.

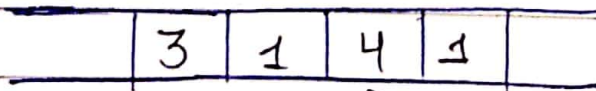


#### Example:

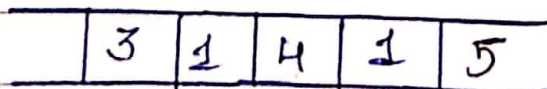
front = 4  
Rear = 6



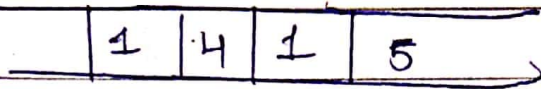
→ Perform Enque front(3)



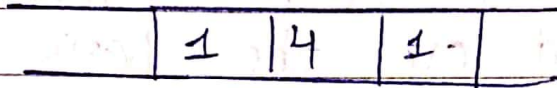
→ Perform Enque rear(5)



→ Perform Deque front will delete 3

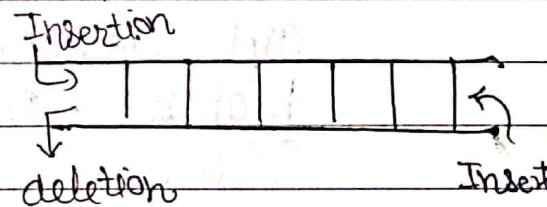
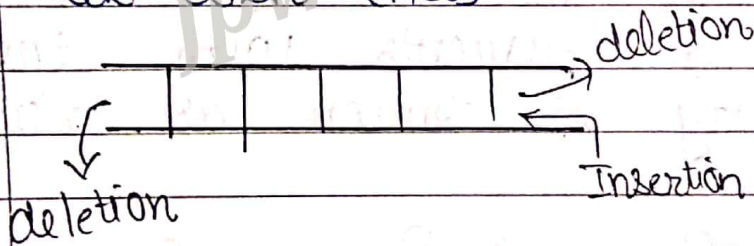


→ Performing Dequeue rear will delete 5.



### Variations

- |   |  |
|---|--|
| ↓   | ↓  |
| Input Restricted Queue                    | Output Restricted Queue                    |
| → Insertion at one end (rear)             | → deletion at one end (front)              |
| → deletion can be performed at both ends. | → Insertion can be performed at both ends. |

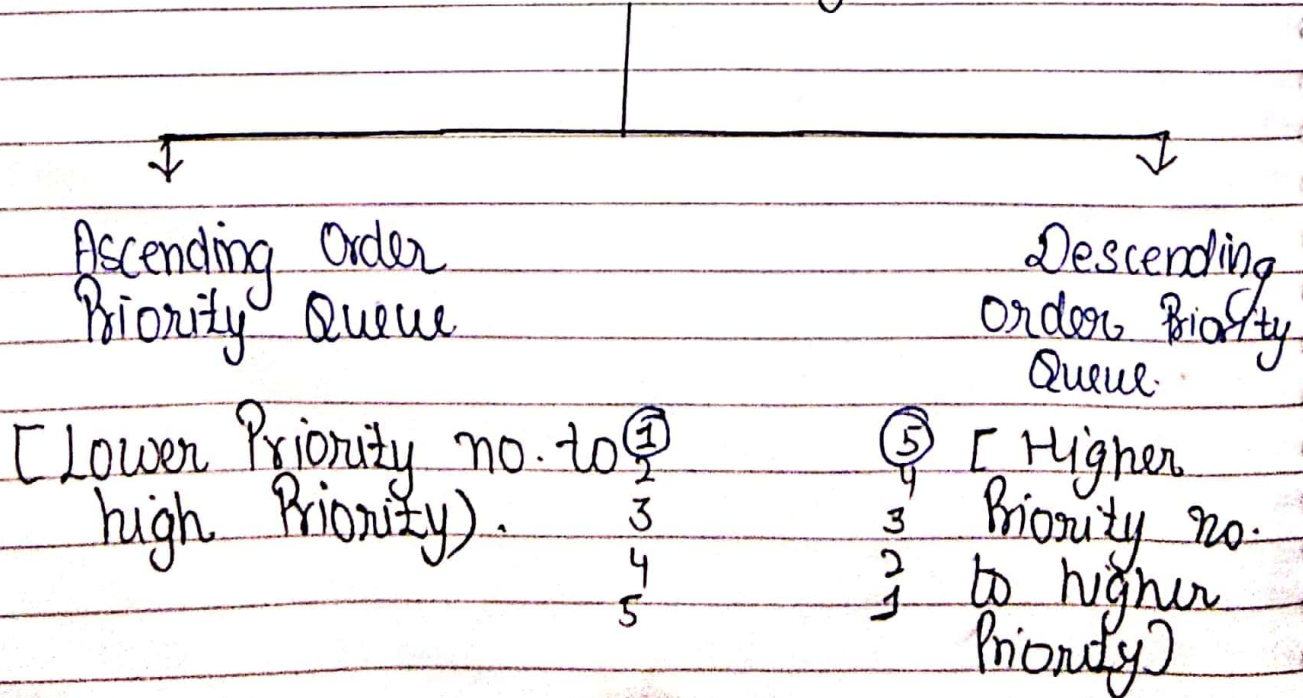


### ④ Priority Queue:-

- In this, Each element is inserted or deleted on the basis of their priority.
- High Priority > Lower Priority
- Same Priority [FCFS basis]
  - ↓
  - first come first serve.

Definition:- A data structure that supports efficient insertions of new element and deletion of elements with the highest priority is known as Priority Queue.

### Types of Priority Queue.



# \* Representation of a Priority Queue :-

INFO	PRIORITY	LINK
222	2	6
444	4	4
555	4	9
333	1	1
111	2	3
666	5	8
777	4	

